Revisited Concept Exercise for Module 4 – No. 1
Monday, April 7, 2014

- **Perform the following signed conversions:**

R(0 1 0 0 0 0)$_2$ → SM(010000 **)$_2$ → DR(** 010000 **)$_2$**

R(1 0 1 1 1 1)$_2$ → SM(110001 **)$_2$ → DR(** 101110 **)$_2$**

0 1 0 0 0 1

SM(1 0 1 1 1 1)$_2$ → DR(110000 **)$_2$ → R(** 110001 **)$_2$**

0 0 1 1 1 1

- **Radix addition (base 2)**

0 1 1 0 1

```
  1 0 0 1 1   -13          1 0 0 1 1
+ 0 1 1 1 1 → +15        + 0 1 1 1 1
-----------              -------------
                       1 0 0 0 1 0
```

(00010)$_2$ → +2

no overflow

```
  0 1 1 1 1   +15          0 1 1 1 1
+ 1 0 0 0 0 → -16        + 1 0 0 0 0
-----------              -------------
                       0 1 1 1 1 1
```

(11111)$_2$ → -1

no overflow

- **Radix subtraction (base 2)**

0 1 1 1 1

```
  1 0 0 0 1   -15          1 0 0 0 1
- 1 1 1 1 1 → -1           0 0 0 0 0
-----------              +         1
                         -----------
                       0 1 0 0 1 0
```

(10010)$_2$ → -14

no overflow

```
  0 1 0 1 1   +11          0 1 0 1 1
- 0 0 0 0 0 → 0            1 1 1 1 1
-----------              +         1
                         -----------
                       1 0 1 0 1 1
```

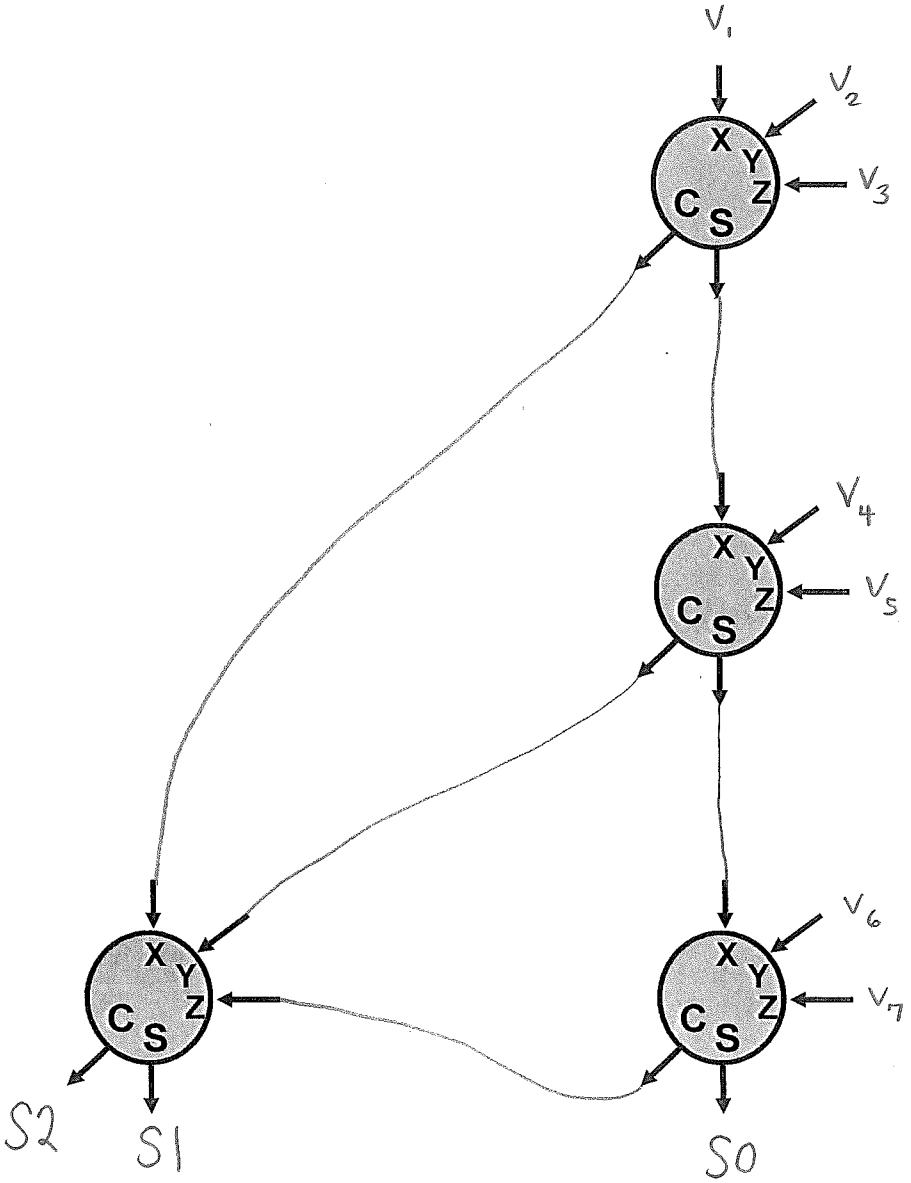
(01011)$_2$ → +11

no overflow

Revisited Concept Exercise for Module 4 – No. 2
Wednesday, April 9, 2014

Tired of writing the names of those you want "kicked off the island" on cards, you wish to modernize the voting scheme used on *Digital Survivor.* Specifically, you would like to design a circuit that tabulates the "stay on the island"/"kick off the island" (yes/no) votes of up to 7 fellow contestants. Armed with only a bucket of full-adder cells (plus 7 switches, some resistors, 3 LEDs, a breadboard, some wires, a power supply, and a large pepperoni pizza), you start your *Incredible Journey.* Assume the tribunal can correctly interpret 3-bit binary numbers.

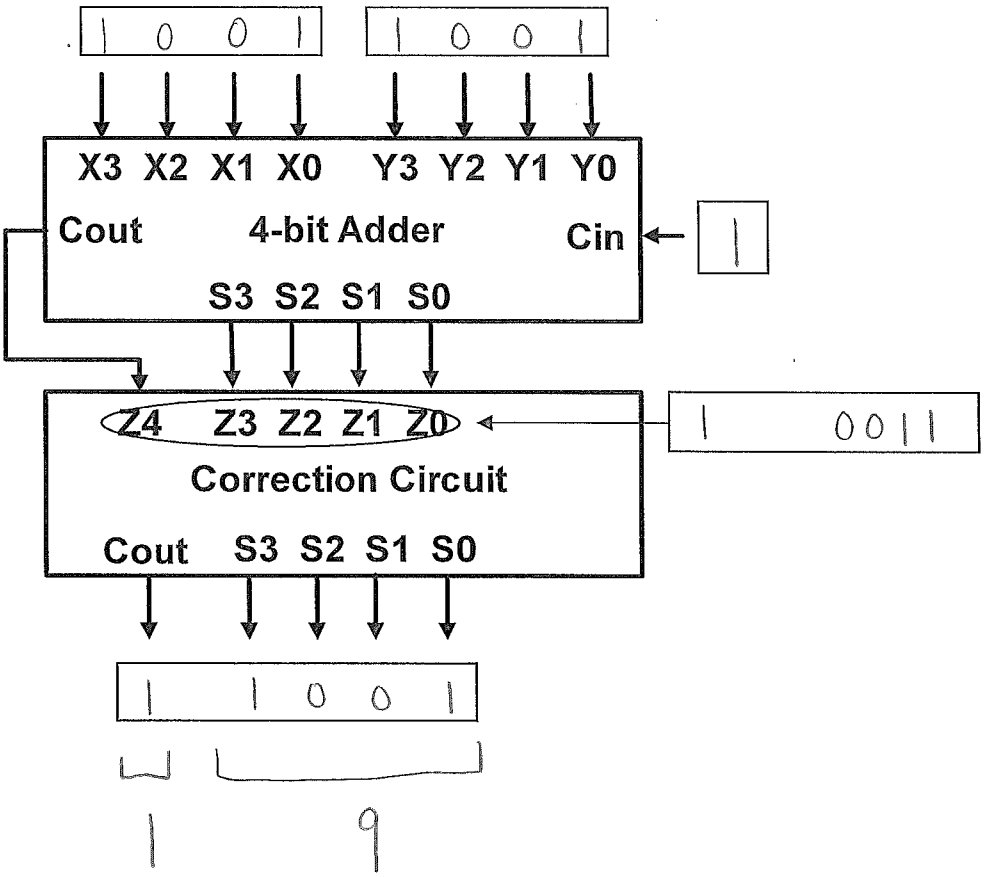
HINT: This is sometimes called a "population counting" circuit – here we simply want to tabulate the number of inputs that are "1". Test your understanding of the operation of the full-adder!

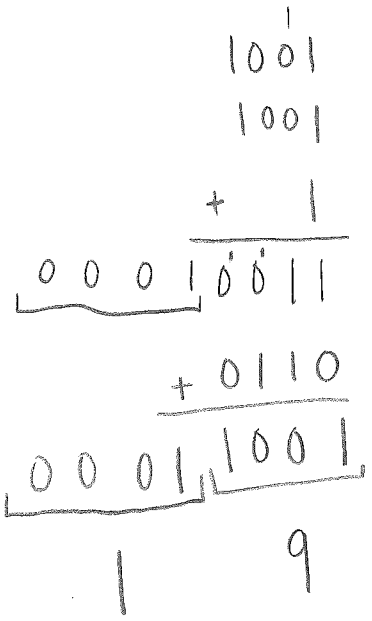


$(111)_2$ in mod.

Revisited Concept Exercise for Module 4 – No. 3
Monday, April 14, 2014

The maximum value that can be generated by a BCD full adder cell is 19_{10}. Write the binary values (bit patterns) that depict this case.





Revisited Concept Exercise for Module 4 – No. 4a
Wednesday, April 16, 2014

1. Write a definition for computer:

   device that sequentially executes a stored program

2. In a "two address" machine, the operands are obtained from:

   register + memory

3. A collection of two or more flip-flops with a common purpose and clocking signal is called a:

   register

4. The two basic steps required to perform an instruction are:

   fetch + execute

5. The component of a computer that keeps track of which instruction to execute next is the:

   program counter (PC) or instruction pointer (IP)

6. The component of a computer that performs arithmetic and logical operations is the:

   arithmetic logic unit (ALU)

Revisited Concept Exercise for Module 4 – No. 4b
Wednesday, April 16, 2014

1. If two additional address bits were added to the Simple Computer, the *number of memory locations* the machine could access would increase:
   (A) by two locations
   (B) by four locations
   (C) by two times the original number of locations
   (D) by four times the original number of locations

2. The condition code bits are generated based on the "D" inputs to the accumulator register flip flops rather than on its "Q" outputs of the accumulator register because:
   (A) we want the flags loaded into the condition code bits to reflect the arithmetic result loaded into the accumulator
   (B) we want to reduce the load on the "Q" output bits
   (C) the "Q" output signals are not available
   (D) the "D" input signals are not inverted

3. The condition code bits need to be latched because:
   (A) the state of the accumulator changes on each clock cycle
   (B) a new arithmetic or logical result is computed on every clock cycle
   (C) condition code bits that are not affected by a certain operation should stay in the same state
   (D) when the AOE signal is negated, the state of the condition code bits might change if they were not latched

4. The increment of the program counter (PC) needs to occur as part of the "fetch" cycle because:
   (A) if it occurred on the "execute" cycle, the new value would not be stable in time for the subsequent "fetch" cycle
   (B) if it occurred on the "execute" cycle, it would not be possible to execute a "JUMP" instruction
   (C) if it occurred on the "execute" cycle, it would not be possible to read an operand from memory
   (D) if it occurred on the "execute" cycle, it would not be possible to read an instruction from memory

5. The program counter (PC) can be incremented on the same cycle that its value is used to fetch an instruction from memory because:
   (A) the synchronous actions associated with the IRL and PCC control signals occur on different phases of the fetch cycle
   (B) the IRL and PCC control signals are not asserted simultaneously by the Instruction Decoder and Microsequencer
   (C) the load of the instruction register (with the value pointed to by the PC) is based on the data bus value prior to the system CLOCK edge, while the increment of the PC occurs after the CLOCK edge
   (D) the load of the instruction register (with the value pointed to by the PC) occurs on the negative CLOCK edge, while the increment of the PC occurs on the positive CLOCK edge

Revisited Concept Exercise for Module 4 – No. 6a
Wednesday, April 23, 2014

1. If output ports bits are ***not*** latched, the data value will ***only*** be output:
   (A) during the fetch cycle of the OUT instruction
   (B) during the execute cycle of the OUT instruction
   (C) when the CLOCK signal is low
   (D) when the CLOCK signal is high

2. Complete the ABEL file, below, that implements a latched input/output port on the Simple Computer. ***Note the port address specified.***

```
MODULE iol

TITLE    'Input/Output Port 10110 - With Output Latch'

DECLARATIONS
DB0..DB7 pin istype 'com';    " data bus
AD0..AD4 pin;                 " address bus
IN0..IN7 pin;                 " input port
OUT0..OUT7 pin istype 'com'; " output port
IOR pin; " Input port read
IOW pin; " Output port write

" Port select (PS) equation for port address 10110

PS = AD4 & !AD3 & AD2 & AD1 & !AD0 ;

EQUATIONS

[DB0..DB7] = [IN0..IN7] ;

[DB0..DB7].oe = IOR & PS ;

" Transparent latch for output port

[OUT0..OUT7] = !(IOW&PS)&[OUT0..OUT7]#IOW&PS&[DB0..DB7];

END
```

Revisited Concept Exercise for Module 4 – No. 6b
Wednesday, April 23, 2014

To implement transfer-of-control instructions in the simple computer requires modification of the program counter.

Briefly describe the modification that is necessary:

The Program Counter (PC) needs to be able to be loaded from the Instruction register (IR) via the address bus.

Complete the ABEL file below that implements the modified program counter:

```
MODULE pcwl
TITLE  'Program Counter with Load Capability'

DECLARATIONS
CLOCK pin;
PC0..PC4 pin istype 'reg_D,buffer';
PCC pin; " PC count enable
ARS pin; " asynchronous reset (connected to START)
POA pin; " PC output on address bus tri-state enable
PLA pin; " PC load from address bus enable
" Note: Assume PCC and PLA are mutually exclusive

EQUATIONS
"          retain state      load          count up by 1
PC0.d = !PCC&!PLA&PC0.q # PLA&PC0.pin # PCC&!PC0.q;

PC1.d = !PCC&!PLA&PC1.q # PLA&PC1.pin # PCC&(PC1.q$PC0.q);

PC2.d = !PCC&!PLA&PC2.q # PLA&PC2.pin # PCC&(PC2.q$(PC1.q&PC0.q));

PC3.d = !PCC&!PLA&PC3.q # PLA&PC3.pin # PCC&(PC3.q$(PC2.q&PC1.q&PC0.q));

PC4.d = !PCC&!PLA&PC4.q # PLA&PC4.pin # PCC&(PC4.q$(PC3.q&PC2.q&PC1.q&PC0.q));


[PC0..PC4].oe = POA;

[PC0..PC4].ar = ARS;

[PC0..PC4].clk = CLOCK;

END
```